

# Quantum Computing: Shor's Algorithm

Feng Siyuan, Liu Chang, Wang Taolei

August 2018

## Contents

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Realization</b>	<b>3</b>
2.1	ExpMod . . . . .	3
2.2	Quantum Order Finding Algorithm . . . . .	5
2.3	The quantum Fourier transform . . . . .	7
2.4	Phase estimation . . . . .	8
<b>3</b>	<b>Conclusion</b>	<b>9</b>

# 1 Introduction

With the development of computability theory, many important problems in computer science and mathematics exist for which there is no known polynomial-time algorithm. The physicist Richard Feynman seems to have been the first to observe that quantum mechanics might allow for faster computation than would be allowed on a classical Turing machine, and thus raised the possibility that a quantum mechanical computer might allow for more robust solutions to problems that have long plagued classical computer science. Among the problems for which no known polynomial-time deterministic algorithm exists is the integer factorization problem, which is thought to be difficult enough to compute that it forms the basis of most of modern cryptographic systems. Shor's algorithm, which this paper provides an exploration of, provides an efficient polynomial time algorithm that operates on a quantum computer. Shor's Algorithm is an important algorithm of quantum computing, which is used for integer factorization. In this article we would like to present an implementation of Shor's Algorithm using  $Q\sharp$ .

## 2 Realization

### 2.1 ExpMod

We tried to use quantum method to realize ExpMod.[1]

The ExpMod is aim to calculate:

$$f(r) = x^r \bmod n,$$

where x is the seed, r is the integer state, and n is the semi-prime number to be factored. Using the fact that r may be written in binary form as

$$r = 2^0 r_0 + 2^1 r_1 + \dots + 2^{N'-1} r_{N'-1},$$

where  $N'$  is the number of qubits of r and  $r_j \in \{0, 1\}$ , we obtain the following expression

$$f(r) = x^{2^0 r_0 + 2^1 r_1 + \dots + 2^{N'-1} r_{N'-1}} \bmod n.$$

This can be mathematically written as

$$f(r) = (\dots ((1 \times x^{2^0 r_0}) \bmod n \times x^{2^1 r_1}) \bmod n \times \dots \times x^{2^{N'-1} r_{N'-1}}) \bmod n.$$

What the gate need to do is to calculate

$$(Number) \rightarrow (Number \times (x^{2^j})^{r_j}) \bmod n,$$

So, we first build the CCNOT gate based on CNOT gate[1].

Then we build Sum gate[2] and Carry gate[3] based on CCNOT and CNOT.

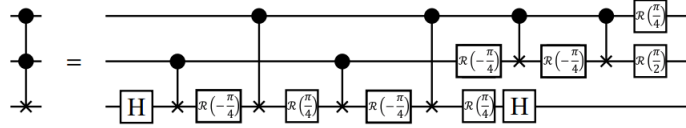


Figure 1: Logic circuit of the CCNOT gate.

After this, we build the Adder gate[4] based on Sum gate and Carry gate.

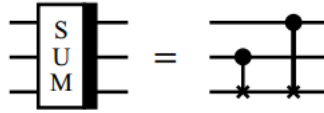


Figure 2: Logic circuit of the Sum gate, which performs to sum the two qubits together.

Based on Adder gate, we build the AddMod gate[5].

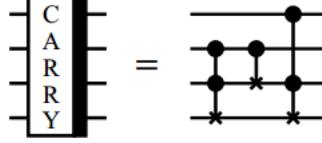


Figure 3: Logic circuit of the Carry gate, which performs to calculate the carry of two qubits.

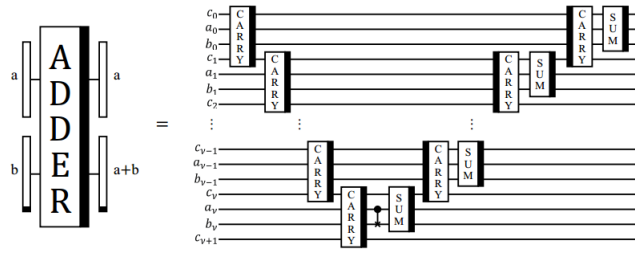


Figure 4: Logic circuit of the Adder gate, which performs to calculate the total sum of two qubit arrays.

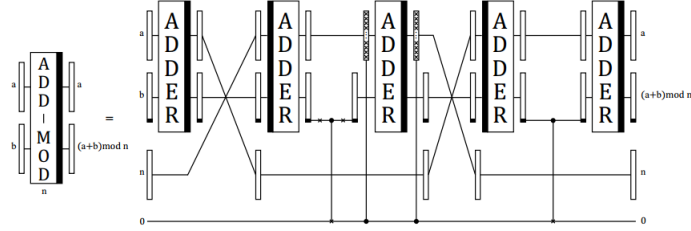


Figure 5: Logic circuit of the AddMod gate, which performs to residue the sum of two qubits by  $n$

After we have all above, we can build CtrlMultiMod gate[6] to calculate

$$(Number) \rightarrow (Number \times (x^{2^j})^{r_j}) \bmod n$$

Finally our ExpMod[7] is like this below.

We build all these gate above just like the circuit shown, sadly some mistakes appear in the interface of gates and it cannot run correctly. We can ensure this method is correct, so we are still fixing it. Meanwhile we also make a ExpMod gate by classic method using Fast power, it can run well.

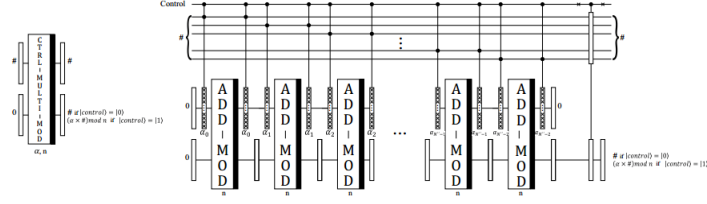


Figure 6: Logic circuit of the CtrlMultiMod gate.

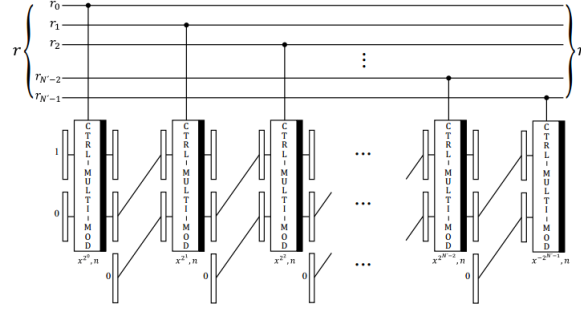


Figure 7: Logic circuit of the modified EXP-MOD gate.

## 2.2 Quantum Order Finding Algorithm

Quantum Order Finding Algorithm is used to solve the Shor's Algorithm much faster.

We first regist  $r$  is the order of  $x$  with regard to  $N$ , then we can build a unitary operate:

$$U|y\rangle = |xy \pmod N\rangle$$

with  $y \in \{0, 1\}^L$ .

A calculation shows that the states defined by

$$|u_s\rangle \equiv \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] |x^k \pmod N\rangle,$$

for interger  $0 \leq s \leq r-1$  are eigenstates of  $U$ , since

$$U|u_s\rangle = \frac{1}{\sqrt{r}} \sum_{k=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] |x^{k+1} \pmod N\rangle = \exp\left[\frac{2\pi i s}{r}\right] |u_s\rangle,$$

So

$$\frac{1}{r} \sum_{k=0}^{r-1} \left( \sum_{s=0}^{r-1} \exp\left[\frac{-2\pi i s k}{r}\right] \right) |x^k \pmod N\rangle = \frac{1}{r} \cdot r |x^0 \pmod N\rangle = |1\rangle.$$

Therefore, using the phase estimation procedure allows us to obtain, with high accuracy, the corresponding eigenvalues  $\exp(2\pi is/r)$ .

### 2.3 The quantum Fourier transform

The quantum Fourier transform on an orthonormal basis  $|0\rangle, \dots, |N-1\rangle$  is defined to be a linear operator with the following action on the basis states

$$|j\rangle \rightarrow \frac{1}{\sqrt{N}} \sum_{k=0}^{N-1} e^{2\pi i j k / N} |k\rangle$$

With a little algebra the quantum Fourier transform can be given the following useful product representation:

$$|j_1, \dots, j_n\rangle \rightarrow \frac{(|0\rangle + e^{2\pi i 0 \cdot j_n} |1\rangle)(|0\rangle + e^{2\pi i 0 \cdot j_{n-1} j_n} |1\rangle) \dots (|0\rangle + e^{2\pi i 0 \cdot j_1 j_2 \dots j_n} |1\rangle)}{2^{\frac{n}{2}}}$$

Then we have the Efficient circuit for the quantum Fourier transform[8].

Where

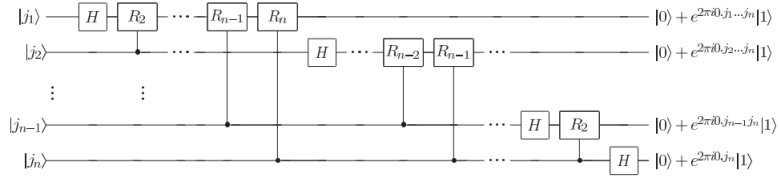
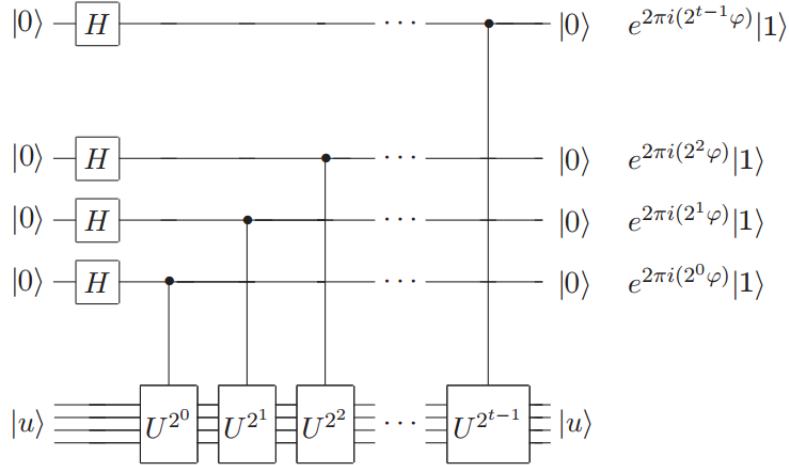


Figure 8: Efficient circuit for the quantum Fourier transform.

$$R_k \equiv \begin{bmatrix} 1 & 0 \\ 0 & e^{2\pi i / 2^k} \end{bmatrix}$$

## 2.4 Phase estimation



In this procedure we use the circuit above to estimate the phase  $\phi$ . The quantum phase estimation procedure uses two registers. The first register contains  $t$  qubits initially in state  $|0\rangle$ , where  $t$  is set with regard to the precision we demand. The second register begins in the state  $|1\rangle$ , which is  $L$  qubits.

Another usage of the phase estimation procedure is to apply the inverse quantum Fourier transform on the first register of the first usage.

The final usage of the phase estimation algorithm is to read out the state of the first register by doing a measurement in the computational basis. When applying the controlled- $U_x^{2^j}$  to the qubits, operate  $U_x$  on  $|1\rangle$  for  $2^j$  times is very time consuming. Therefore, we can calculate the 2 to the  $k^{th}$  power in advance instead of applying the controlled- $U_x^{2^j}$  gate  $2^k$  times.



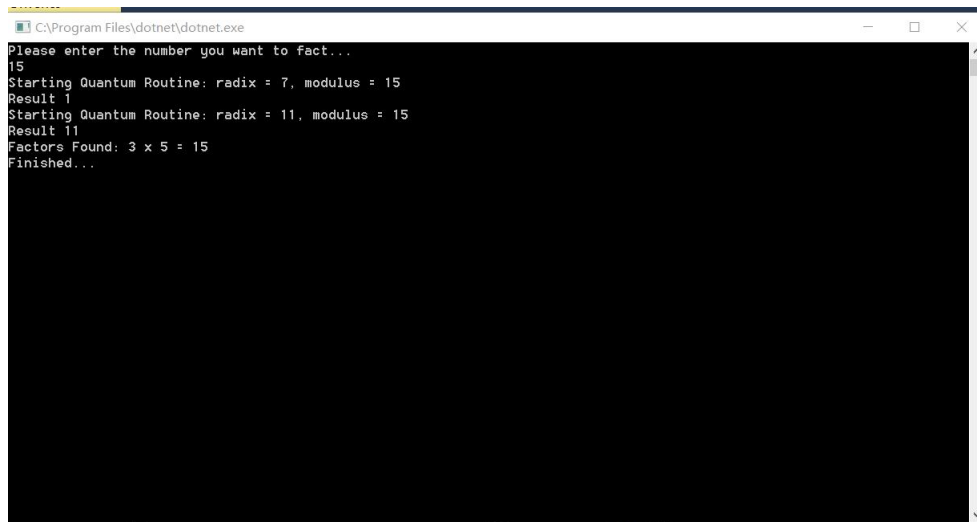
### 3 Conclusion

We do some basic tests of our program , which take less than ten minutes on my laptop:

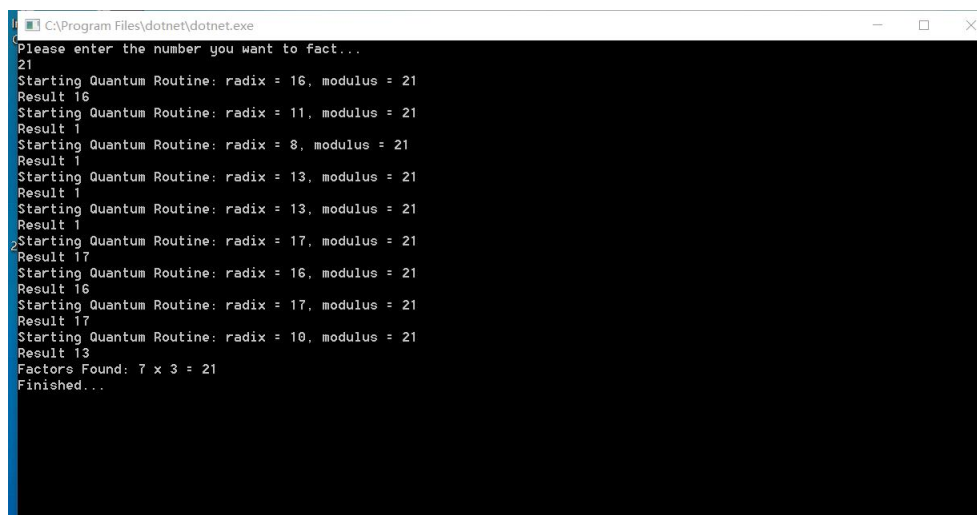
$$15 = 3 \times 5$$

$$21 = 3 \times 7$$

Actually, we want to try some bigger numbers like 33 , but our program runs so slow that the laptop cannot hold it.

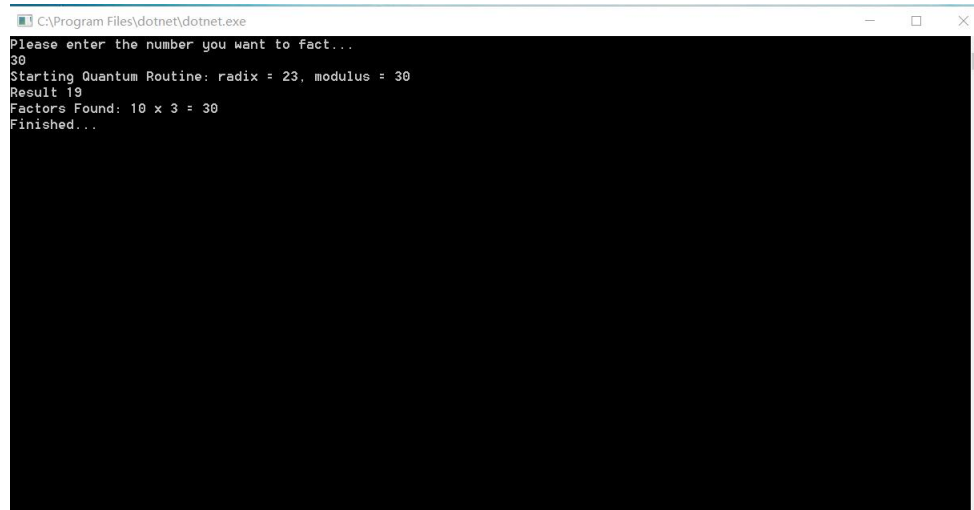


```
C:\Program Files\dotnet\dotnet.exe
Please enter the number you want to fact...
15
Starting Quantum Routine: radix = 7, modulus = 15
Result 1
Starting Quantum Routine: radix = 11, modulus = 15
Result 11
Factors Found: 3 x 5 = 15
Finished...
```



```
C:\Program Files\dotnet\dotnet.exe
Please enter the number you want to fact...
21
Starting Quantum Routine: radix = 16, modulus = 21
Result 16
Starting Quantum Routine: radix = 11, modulus = 21
Result 1
Starting Quantum Routine: radix = 8, modulus = 21
Result 1
Starting Quantum Routine: radix = 13, modulus = 21
Result 1
Starting Quantum Routine: radix = 13, modulus = 21
Result 1
Starting Quantum Routine: radix = 17, modulus = 21
Result 17
Starting Quantum Routine: radix = 16, modulus = 21
Result 16
Starting Quantum Routine: radix = 17, modulus = 21
Result 17
Starting Quantum Routine: radix = 10, modulus = 21
Result 13
Factors Found: 7 x 3 = 21
Finished...
```

By the way, if we ignore the condition of Shor's Algorithm, we can use our program to fact 30:



```
C:\Program Files\dotnet\dotnet.exe
Please enter the number you want to fact...
30
Starting Quantum Routine: radix = 23, modulus = 30
Result 19
Factors Found: 10 x 3 = 30
Finished...
```

## References

- [1] Y. S. Nam. *Running Shor's Algorithm on a complete, gate-by-gate implementation of a virtual, universal quantum computer*. Middletown, Connecticut, 2011.